# AES Candidates: A Survey of Implementations

Helger Lipmaa[1]

Küberneetika AS; Akadeemia 21, 12617 Tallinn, Estonia
helger@cyber.ee

**Abstract.** We present a cross-table of almost all publicly known implementations of AES candidates, including the ones done by the authors. A short overview of our own implementations of Rijndael is given. The relative easiness of doing "the world best" implementations and a lot of gaps in the table force us to ask if there is enough information known to really decide which ciphers are fast and which are not. (This paper only compares the *encryption* speed in the case of the 128-bit keys.)
In the conclusions we present a very brief survey of the known attacks to the candidates, stressing the fact that other aspects of the candidates are still less known (at least to the public). We finish this paper with apparent conclusions that the first round of the AES process has been too short, but we still give recommendations which candidates should be elected to the second round basing on the knowledge known to the public at the moment of writing this paper.

## 1 Introduction

The author of this paper has been keeping a cross-table [Lip99] for all[1] (the known) implementations of the AES candidates since the July 1998, receiving a huge feedback for a number of persons. This experience has taught us that there exist pairs (processor,candidate), for what no implementations are known. Most notable was the lack of the Pentium II assembler numbers of DEAL, MARS and Rijndael, but there were (and are) also many missing implementations on the Pentium II Borland C++ (the NIST standard platform). In an vain attempt to fill up some gaps, we tested Brian Gladman's implementations on a UltraSPARC II machine. Thereafter we concentrated on finishing Rijndael implementations for several platforms. Our three implementations (done during three days) are the best known for the corresponding (processor,language) pair, and yet not fully optimized. This fact stress, as we believe, the incompleteness of the current table and therefore also the fact the available information is not sufficient to answer reliable to the question which ciphers are fast and which are not.

Still, there is something that can be said based on this table. There are some ciphers that are extremely fast and others that are relatively slow. There are some ciphers that are fast on every platform and others that are optimized for the AES test platform (Pentium II) and therefore somewhat slower on other types of processors. We will state our knowledge by providing some tables. Thereafter we focus on our own implementations of Rijndael, which is a cipher of our choice for several completely different reasons.

[1] Note that we do not discuss Magenta and Loki97 that were broken before the end of the AES1 Conference in August 1998.

Later we give a very brief overview of the known attacks against all the candidates, stressing the fact that other aspects of the candidates are still less known (at least to the public). We finish this paper with apparent conclusions that the first round of the AES process has been too short and give recommendations which candidates should be elected to the second round basing on the knowledge known to the public at the moment of writing this paper.

## 2 The Implementation Cross-Table

### 2.1 Explanation to The Table

The cross table of tuples (processor,compiler,candidate cipher) for (almost) all known implementations is presented in Table 1. The entries are of form "*x* C (I)", where *x* is the number of cycles per 128-bit block, C is the optional compiler and "(I)" is the optional implementer. For example, in the row "Pentium II/BC++ 5.0" and in the column "E2" the entry 711 means that the best known implementation of E2 on the Pentium Pro/Pentium II (using the BC++ 5.0 compiler) takes 711 cycles to encrypt one block.

There are some exceptions. Brian Gladman's[2] original implementations (optimized for the Pentium II) have been ported to Pentium (by Brian himself from an old version of his source code), UltraSPARC-II (by Helger Lipmaa, using Sun C WorkShop 5.0), to the Alpha AXP 21164 ([Gra99], using DEC CC and gcc 2.8.1), to Hewlett-Packard PA7000 [Gra99], to the Alpha AXP 21264 ([Gra99], DEC CC). We decided to cite all these ports. Our feeling is that it will help to grasp how much does it cost just to port a cipher to another platform. (As seen by comparing entries in those rows the speed ratio of some ciphers depends very heavily on the processor used). There's also a separate row is for Kenneth Almquist's [Alm99] *estimations* of speeds at AXP 21164.

### 2.2 Quoted Numbers

Most of the estimates are still from the original papers submitted to the First AES Candidate Conference. Exceptions are:

**DFC** The Alpha C implementations are from [Har99]; the C implementations for UltraSPARC and SPARC 170 are by Fabrice Noilhan. The P2 assembler implementation is by Terje Mathisen.

**E2** Some numbers are from its homepage.

**RC6** The assembly implementation is by Ted Krovetz. Some of the other data have been taken from its homepage.

**Rijndael** Rijndael's Pentium II assembler, Pentium II (egcs) and UltraSPARC-II (Sun C) implementations are by Helger Lipmaa.

[2] The data given in Table 1 differs somewhat from [Gla99]. The main reason is that the data here also includes the endianness conversion times. Another difference is that I only provide 128-bit key *encryption* times, not mean of encryption and decryption. We have performed no check of the correctness of his or any other parties implementations. Our numbers are based on the (hopefully justified) trust in the implementers.

| Machine/ compiler | CAST-256 | Crypton | DEAL | DFC | E2 | Frog | MARS | RC6 | Rijndael | Safer+ | Serpent | Twofish | HPC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **32-bit software** | | | | | | | | | | | | | |
| **Pentium Pro/Pentium II** | | | | | | | | | | | | | |
| BC++ 5.0 | 1790 | | | | 711 | 2600 | 920 | 616 | ≈ 775 | | 1738 | 640 | 3500 |
| MSVC | | 475 (5.0) | | 2432 (4.0) | | | | ≈262 (DS 97) | | 2080 (5.0) | | | |
| DJGPP/ egcs/ gcc | | | | | | | 390 (djgpp) | | 370 egcs, [Lip99] | | | | |
| assembly | 815 | 390 | | 480 (TM) | 415 | | | 243 (TK) | 282 [Lip99] | | | 258 | |
| [Gla99] MVC++ 6.0 | 668 | 478 | 2339 | 1688 | 734 | 2572 | 376 | 270 | 374 | 1746 | 992 | 378 | 1468 |
| **Pentium** | | | | | | | | | | | | | |
| MVC++ 6.0, * | 1131 | 816 | | 3172 | 1079 | | 760 | 758 | 702 | 2449 | 1279 | 673 | |
| assembly | | | | | | | | | 320 | | | 290 | |
| EGCS 1.0.2 | | | | | | | | | 950 | | | | |
| **Alpha AXP 21164** | | | | | | | | | | | | | |
| DEC CC | | | | 323 [Har99] | | | | | | | | | ≈420 |
| gcc | | | | 366 [Har99] | | | | | 490 (LG) | | | | |
| DEC CC, * | 749 | 499 | 2752 | 1230 | 679 | 2752 | 507 | 559 | 516 | 1502 | 998 | 490 | 930 |
| gcc 2.8.1, * | 979 | 593 | 2928 | 1946 | 813 | 4197 | 771 | 625 | 617 | 3347 | 1444 | 511 | 3372 |
| asm | | | | | 587 | | | | | | | | 600 |
| asm [Alm99] | 600 | 408 | 2528 | 304 | 471 | | 478 | 467 | 340 | 656 | 915 | 360 | 380 |
| **SPARC** | | | | | | | | | | | | | |
| Sun C 5.0, * | 694 | 477 | 2781 | 2692 | 711 | 2337 | 840 | 1161 | 334 | 3002 | 996 | 487 | 1465 |
| Ultra-SPARC C | 1180 | 615 | | 910 | | | | | 328 [Lip99] | | | 750 | 450 [Gra99] |
| Sparc 170 C | | | | 969 | | | | | | | | | |
| Sparc 170 asm | | | | 802 | | | | | | | | | |
| **Other (RISC, 680x0, ...)** | | | | | | | | | | | | | |
| HP PA7000, * | 1275 | 865 | 3940 | 2435 | 990 | 2620 | 950 | 1085 | 735 | 5085 | 1345 | 755 | 1315 |
| AXP 21264, * | 615 | 353 | 2010 | 232 [Har99] | 510 | 3750 | 450 | 382 | 285 | 929 | 855 | 315 | 420 (LG) |
| PPC 604e/750 C | | | | | | | 300 | | | | | 590 | |
| Strong-ARM asm | | | | 560 [Har99] | | | | | | | | | |
| 68040 C | | | | | | | | | | | | 3500 | |

**Table 1.** The Cross-Table of All Known Implementations.

**Safer+** The Borland C++ implementation data is taken from the recent posting of Lily Chen on http://aes.nist.gov where the speed was stated to be 33 Mbit/s.

**Twofish** Assembly data is taken from a posting of Doug Whiting on http://aes.nist.gov.

**One-man implementations** Already mentioned separate rows for [Gla99], its ports (marked with "*") and for [Alm99].

We left the hardware estimations and the timings for 8-bit processors (a survey of which is present in [Lip99] and in [SKW+99]) as still yet very incomplete (we have been unable to obtain estimations for 7 ciphers out of 13!). We would like to stress once more that what "we know" is not equal to "what exists." The entries in Table 1 should mostly be taken just as upper bounds (or lower bounds, in the case of [Alm99]) for implementations.

## 3 Sorted Table

Table 1 contains all the data we were able to obtain and it is far from being perfect. It is certainly not complete. It is not comprehensive. But still, we can draw some conclusions. Namely, at least one implementation of every candidate is known for at least three different platforms: Intel's Pentium II, Sun's UltraSPARC and Compaq's Alpha AXP 21164[3]. The numbers corresponding to the *best* known implementations on the three platforms is presented in Table 2.

| Cipher | Pentium II | Alpha | SPARC | $\bar{x}$ |
|---|---|---|---|---|
| Rijndael | 284 | 490 | 328 | 367.3 |
| Twofish | 258 | 490 | 487 | 411.7 |
| Crypton | 390 | 499 | 477 | 455.3 |
| DFC | 480 | 323 | 802 | 535.0 |
| E2 | 415 | 587 | 711 | 571.0 |
| MARS | 376 | 507 | 840 | 574.3 |
| RC6 | 243 | 559 | 1161 | 654.3 |
| CAST-256 | 668 | 749 | 694 | 703.3 |
| HPC | 1468 | $\approx$ 420 | 450 | 779.3 |
| Serpent | 992 | 998 | 992 | 994.0 |
| Safer+ | $\approx$ 775 | 1502 | 3002 | 1759.7 |
| Frog | 2572 | 2752 | 2337 | 2553.7 |
| DEAL | 2339 | 2752 | 2781 | 2624.0 |

**Table 2.** Speeds of the best known implementations of 13 AES candidates on the Pentium II, the Alpha and the UltraSPARC. The candidates are ordered by the average number over the three platforms.

To get a "fair" representation of the efficiency of different candidates, we sorted Table 2 in the order of the average value of these three implementations. There are

[3] We omitted PA7000 and AXP 21264 because these platforms did not have any implementations by authors. Inclusion of this data would not have changed the order significantly.

several reasons for that. At first, while by far the most popular processor at this moment is the Pentium II, it will with high probability not be the standard after five years. Intel's plans to supersede the Pentium family with the new 64-bit architecture IA64 have been known to the public for a while. While the Alpha and the SPARC architectures are not as popular as the Pentium II, they can be seen as (typical) representatives of the 64-bit processors and therefore should definitely be accounted. On the other hand, inclusion of the Pentium II is still a must not only because of it's popularity but also because it is necessary to compare the performances on completely different architectures — the select AES cipher has to be efficient on machines having different endianness and word length.

Let us state some evident conclusions. At first, performance of some ciphers is almost the same on the Alpha AXP 21164 and UltraSPARC. It is a consequence of the usage of the same set of sources [Gla99] but also from the many similarities between the Alpha and SPARC as 64-bit RISC architectures. Twofish, Crypton, CAST-256, HPC, Serpent, Frog and DEAL fall into this class. For some ciphers, Alpha performance is much better. In the case of Rijndael, the difference is probably due to the bad code (Rijndael should be at least $1.2$ times faster than Crypton, i.e., $< 415$ cycles on Alpha; by the same reasoning Crypton we estimate a hand-optimized implementation of Crypton on the Pentium II to take about 340 cycles). In the case of DFC, a lot of work [Har99] has been done to achieve fast implementations on Alpha. MARS and RC6 suffer on the UltraSPARC due to the their heave use multiplication and data-dependent rotation (cf [SKW+99]).

Comparing the Pentium II to 64-bit architectures one notices immediately that some ciphers (RC6, MARS) are optimized for the Pentium II and that some (DFC, HPC) are optimized for 64-bit architectures, while performances of the majority of ciphers are similar on all architectures.

Thus, some ciphers that are very fast on very architecture: Rijndael, Twofish, Crypton, DFC, E2 and MARS. Three ciphers are slow on every architecture: Safer+, Frog and DEAL. Others are in between. Including RC6 (contrary to the public belief).

## 4 Our Choice: Rijndael

We decided to concentrate our own efforts on Rijndael by several reasons. Of course, it is fast (the fastest in Table 2). It is (as far as we know) the fastest cipher on PA7000 and the second fastest on AXP 21264. It's inner parallelism is almost unlimited (16!). It is suitable for a wide range of architectures due to it's simplicity (it does not use any specific 32- or 64-bit operations, unlike DFC, MARS, RC6, ...), it is likely to be extremely efficient in hardware. While 192-bit and 256-bit key versions of Rijndael take more time due to the added rounds, it is compensated by the designers choice to allow a variety of block lengths. The same implementation that takes 284 cycles per $128/128$ (key/block) Rijndael encryption would take (an estimated) 392 cycles per $256/256$ Rijndael encryption (or 196 cycles per 128-bit block).

Of course, speed alone does not matter. More important is the fact that Square [DKR97] — the cipher Rijndael bases on — has been unbroken for two years, four times the period it has been since most of other ciphers were published. The design

strategy of Square seems to guarantee that the cipher will stay secure (until a completely new kind of cryptographic attack is found), and Rijndael should be much more secure than Square. Although there is a number of other ciphers, against which no significant attack has been mounted (Sect. 5), we feel that only DFC, E2 and Serpent have the same security level. And the third consideration is just the fact that Rijndael is unpatented.

Summarizing the desiderata we found Rijndael to be the best overall candidate to start work on. During this work, we completed three different implementations of Rijndael. The first, a (still not fully) hand-optimized assembler implementation took two days and based on the fact that the Pentium II is good at handling bytes. The second implementation (egcs, Pentium II), took about two hours, and needed a completely different approach due to the sub-optimality of egcs when handling 8-bit data. The (another not fully optimized) result (370 cycles) was better than Brian Gladman's implementation using the commercial MS VC++ compiler (Gladman's implementation took 456 cycles with egcs, for example). The third (even less optimized) implementation took about 20 minutes to finish, and makes 328 cycles on UltraSPARC-II with Sun C 5.0 (336 cycles per Gladman's code).

Our timing routines use the standard C library function `times` around a long loop. We estimate that the inner code of Rijndael takes at least 5 less cycles on the Pentium II if we exclude this overhead.

We expect Rijndael to be fast on other platforms as well. It performs very fast in the smartcards [DR98] and in the hardware [SKW+99].

## 5 Conclusions

We wrote (from scratch) 3 different optimized implementations of Rijndael in about two days that happen also be the best known implementations for the given (processor,language) pair. The code can still be bettered, and it wasn't terrible hard. Just no one had done it. And yet, implementing candidates is at this moment what everyone does. There hasn't been a lot of successful cryptanalysis during last 3-4 months. Which only stresses the fact that the period before the 2nd round is too short.

What is known? Loki, Frog and Magenta and DEAL have been broken. A small attack is known against the MARS key schedule. An attack to Crypton's key schedule and a known weirdness of Crypton's S-boxes (the last has no known cryptographic applications). Thus, there are 9 ciphers with no known attacks (better than those given in submissions). The attacks against MARS and Crypton do not count much. Leaves 11.

From those, Serpent and Safer+ can be eliminated because of their low performance. May be we could use the same reason to eliminate CAST-256. Leaves 8. HPC is slow on the Pentium II, but it is fairly fast on Alpha and UltraSPARC. Don Coppersmith's reasoning on AES Discussion Groups is not sufficiently formal to be a reason for disgracing HPC. Almost the same (to some less extent) is true for DFC: it has average performance on the Pentium II, but it is very fast on Alpha. Don Coppersmith's comments on AES Discussion Groups are not resolute enough.

The mistakes in Crypton, HPC, DFC and MARS are very subtle. So, there are 8 candidates: RC6, Twofish, Rijndael, E2, Crypton, DFC, MARS, HPC. Hard choice.

The first four do not have (yet) *any* later discovered weaknesses, the first three of them are very fast. But there is no *good* reasons to prefer DFC to Crypton or vise versa.

To conclude the paper, we will note a curious fact. None of the AES candidates seems to get any serious benefit from the multimedia architectures (Intel's MMX, Sun's VIS, ...). Some of the candidates (MARS [BCD+98] and RC6 [RRSY98]) rely on the 32-bit unsigned multiplication. The reasoning of the authors is that such multiplication is very cheap on nowadays common microprocessors. This claim is indeed true, but the best known multimedia technologies cannot be used to accelerate these ciphers because of the lack of a 32-bit parallel multiplication. There is a certain tradeoff (and even a contradiction) here. MARS and RC6 are optimized for the new 32-bit processors (mainly for the Pentium II; as it was already noted, these ciphers are far from optimal on other architectures), utilizing fully the 32-bit operations provided by such processors. At the same time, these ciphers ignore the multimedia extensions existing in the very same processors.

Most of the candidates use some form of $S$-boxes and/or lookup tables and do not take major advantage from the multimedia extensions of MMX (though they could benefit from the larger cache or word-size) as the MMX registers cannot be used as memory pointers. Parallelization of these ciphers would need accessing several 'randomly" chosen memory cells simultaneously [Lip98].

## 6 Acknowledgments

Thank you to Kenneth Almquist, Lily Chen, Jim Foti, Brian Gladman, Louis Granboulan, Robert Harley, Terje Mathisen, Fabrice Noilhan, Richard Outerbridge, Richard Schroeppel, Serge Vaudenay, who all have contributed to my effort of keeping the page [Lip99].

## References

[Alm99] Kenneth Almquist. Aes candidate performance on the alpha 21164 processor (version 2). Unpublished, a posting from `sci.crypt`. Information available from `http://home.cyber.ee/helger/aes/kenneth.txt`, January 1999.

[BCD+98] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas Jr., Luke O'Connor, Mohammad Peyravian, David Safford, and Nevenko Zunic. Mars — a candidate cipher for aes. Available at `http://www.research.ibm.com/security/mars.html`, June 1998.

[DKR97] Joan Daemen, Lars Knudsen, and Vincent Rijmen. The block cipher square. In Eli Biham, editor, *Fast Software Encryption '97*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165, Haifa, Israel, January 1997. Springer-Verlag.

[DR98] Joan Daemen and Vincent Rijmen. The block cipher Rijndael. In *Third Smard Card Research and Advanced Applications Conference Proceedings*, 1998. To appear.

[Gla99] Brian Gladman. AES algorithm efficiency. Unpublished. Information available from `http://www.seven77.demon.co.uk/aes.htm`, January 1999.

[Gra99] Louis Granboulan. AES: Analysis of the submissions. Unpublished. Information available from `http://www.dmi.ens.fr/ granboul/recherche/AES.html`, January 1999.

[Har99] Robert Harley. Personal communication. January 1999.

[Lip98] Helger Lipmaa. IDEA: A cipher for multimedia architectures? In *Selected Areas in Cryptography '98*, Lecture Notes in Computer Science, Kingston, Canada, August 1998. Springer-Verlag. To appear.

[Lip99] Helger Lipmaa. AES candidates: A survey of implementations. An on-line table. Information available from `http://home.cyber.ee/helger/aes/`, January 1999.

[RRSY98] Ronald L. Rivest, Matt J. B. Robshaw, R. Sidney, and Y. L. Yin. The rc6 block cipher. Available at `http://theory.lcs.mit.edu/~rivest/rc6.ps`, June 1998.

[SKW+99] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, and Chris Hall. Performance comparison of the AES submissions. Unpublished. Information available from `http://www.counterpane.com`, January 1999.